Homework 5

1. Fitting non-quadratic losses to data. In non_quadratic.json, you will find a 500 × 300 matrix U_train and a 500-vector v_train consisting of raw training input and output data, and a 500 × 300 matrix U_test and a 500-vector v_test consisting of raw test input and output data, respectively. We will work with input and output embeddings x = φ(u) = (1, u) and y = ψ(v) = v. Our performance metric is the RMS error on the test data set.

In regression_fit.jl we have also provided you with a function

\[ \text{regression\_fit}(X, Y, l, r, \lambda) \]

This function takes in input/output data X and Y, a loss function \( l(\hat{y}, y) \), a local regularizer function \( r(\theta) \), and a local regularization hyper-parameter \( \lambda \). It outputs the model parameters \( \theta \) for the RERM linear predictor. You must include the Flux and LinearAlgebra Julia packages in your code in order to utilize this function. You will use this function to fit a linear predictor to the given data using the loss functions listed below.

- Quadratic loss: \( \ell(\hat{y}, y) = (\hat{y} - y)^2 \).
- Absolute loss: \( \ell(\hat{y}, y) = |\hat{y} - y| \).
- Huber loss, with \( \alpha \in \{0.5, 1, 2\} \): \( \ell(\hat{y}, y) = p_{\alpha}^{\text{hub}}(\hat{y} - y) \), where

\[
p_{\alpha}^{\text{hub}}(r) = \begin{cases} 
  r^2 & |r| \leq \alpha \\
  \alpha(2|r| - \alpha) & |r| > \alpha 
\end{cases}
\]

- Log Huber loss, with \( \alpha \in \{0.5, 1, 2\} \): \( \ell(\hat{y}, y) = p_{\alpha}^{\text{log\_hub}}(\hat{y} - y) \), where

\[
p_{\alpha}^{\text{log\_hub}}(r) = \begin{cases} 
  r^2 & |r| \leq \alpha \\
  \alpha^2(1 - 2\log(\alpha) + \log(r^2)) & |r| > \alpha 
\end{cases}
\]

We won’t use regularization so you can use \( r(\theta) = 0 \) and \( \lambda = 1 \) (though your choice of \( \lambda \) does not matter).

Report the training and test RMS errors. Which model performs best? Create a one-sentence conjecture or story about why the particular model was the best one.

Julia hint. You will need to define the loss functions above in Julia. You can do this in a compact (but readable) form by defining the function inline.

For example, for quadratic loss, \( l_{\text{quadratic}}(yhat, y) = (yhat - y)^2 \) and similarly, for huber loss, \( l_{\text{huber}}(yhat, y) = \text{huber}(yhat - y, \alpha) \), where, \( \text{huber}(r, \alpha) \) function would be defined separately.

You will also need to do the same for the regularizer (although it is zero). You can do this with \( r(\theta) = 0 \). Pass the names of the function (say, \( l_{\text{quadratic}} \) instead of \( l \)) in the \( \text{regression\_fit}(X, Y, l, r, \lambda) \) function above.
2. **Non quadratic regularizers.**

Using the same data file `non_quadratic.json` and provided `regresion_fit` function, we will investigate the impact of different regularizers.

Using the **test RMS error**, select the best 2 loss functions from Problem 1. We will evaluate them with the following regularizers:

- Quadratic or ridge regularization: $r(\theta) = \lambda \|\theta_{2:k}\|^2_2$
- Lasso regularization: $r(\theta) = \lambda \|\theta_{2:k}\|_1$
- No regularization (you’ll use this as a baseline)

You are free to choose the weight $\lambda$. A good starting choice is 0.1 but you are encouraged to experiment. Keep in mind different weights may work better for different functions.

(a) For the 2 best loss functions in Problem 1, and the regularizers listed above, report the training and test RMS errors. What is the best loss + regularizer combination? Don’t forget to try a few different values of $\lambda$ (you only have to report the one you choose).

(b) Provide a (brief) comment on whether your results in this problem agree with your results from Problem 1.

(c) Some loss functions, such as quadratic loss, are convex. Others are not (such as log-Huber). Convex functions are advantageous because they can be reliably optimized. Is your best loss + regularizer convex? If not, what is the best convex loss + regularizer you found?

*Julia hint.* Refer question 1.

3. **How often does your predictor over-estimate?** In this problem, you will identify how often linear predictors with tilted absolute losses over-estimate.

In `residual_props.json`, you will find a 500 × 10 matrix `U_train` and a 500-vector `v_train` consisting of raw training input and output data, and a 500 × 10 matrix `U_test` and a 500-vector `v_test` consisting of raw test input and output data, respectively. We will work with input and output embeddings $x = \phi(u) = (1, u)$ and $y = \psi(v) = v$, and use no regularization ($r(\theta) = 0$). You will also use `regression_fit.jl` from the previous problem.

Recall that the tilted absolute penalty is

$$p_\tau(u) = \begin{cases} -\tau u & u < 0 \\ (1 - \tau)u & u \geq 0, \end{cases}$$

where $\tau \in [0, 1]$. Fit a linear predictor to the given data using the tilted absolute penalty, i.e., $\ell(\hat{y}, y) = p_\tau(\hat{y} - y)$, for $\tau \in \{0.15, 0.5, 0.85\}$. For both the training set and the test set, report how frequently this predictor over-estimates (% of $\hat{y} > y$), and plot the empirical CDFs (Cumulative Distribution Function) of the residuals.

*Hint.* A predictor $\hat{y}$ over-estimates $y$ when $\hat{y} > y$. To generate an empirical CDF plot of the residuals $r$ of length $d$, you may plot `collect(1:d)/d` versus `sort(r)`.  